Touch Pattern Recognition for authentication

13.05.2023

—

Diarra Yacouba, Kanté Younouss Koly, Haïdara Mohamed,

Diabaté Maïmouna, Maïga Maïmouna Henna

@Institut Privée Africain de Technologie et de Management (ITMA) : Not implicated in the research

Baco Djicoroni ACI, Bamako, Mali

Baco Djicoroni ACI, Bamako, Mali

Overview

This interim report aims to be a good checkpoint about the progression of this project. The Touch Pattern Recognition project is a project that aspires to revolutionize authentication systems. This project comes from five students at the @ITMA university Diarra Yacouba, Kanté Younouss Koly, Haïdara Mohamed, Diabaté Maïmouna and Maïga Maïmouna Henna as part of the second edition of "TRANSFORMATION NUMERIQUE", the competition organized by Africa_Digital in May 2023.
These students participated in the "TRANSFORMATION NUMERIQUE" universities league which was about Data security.
Note that this file can be edited at any moment due to the ongoing status of the project and that all events mentioned in this document are localized in Bamako/Mali between April 30 and May 13 2023.

With that said, The idea behind the project is to replace traditional system authentication credentials (username/ password) by a machine learning (ML) model trained to Recognize the user of the system through touch data on touch sensitive devices.
For this purpose data are represented with different fields such as positions, speeds, directions, latency (delay) of each touch.

This document will walk through the project, from data acquisition to Model deployment passing through issues and problems encountered.

Goals

This project aims to:

1. Rethink the way we think of authentication systems.
2. Bring more innovation and security into authentication systems.
3. Make access to systems easier by replacing passwords that can be tedious stuff as they are easy to forget or hack.
4. Look for new applications of AI systems that are becoming more and more powerful.

Data acquisition & preprocessing

We are going to talk about the way data has been collected and processed for training.

I. Data Acquisition

Firstly, we utilized an existing mobile web application called "TouchTracker" developed by Marco Cancellieri. This application has a simple user interface that displays a green screen and records all gestures performed on the screen written in JavaScript.
More info at : TouchTracker

However, the application's original purpose was not to collect touch-related data that we need, so we made modifications to its source code to collect the necessary data. Specifically, you modified the application's code to collect touch positions, speeds, directions, durations, and timestamps.

These modifications allowed you to gather detailed information about the touch gestures being performed on the screen, which is essential for training a machine learning model to recognize and make predictions accurately.

Once the data was collected, it was exported in .json file format. This is a structured data format that is easily readable and processed by most programming languages, including Python, the language we used to train your machine learning model.

To begin with the data acquisition process, we created a user interface for our TouchTracker based web. We did this by modifying another source code that was initially written for mouse events. In order to adapt this code to work with touch events, we needed to modify the HTML, CSS, and JS code.

HTML, CSS, and JS are the three foundational technologies used to create websites and web applications. HTML provides the structure of the page, CSS controls the presentation and layout, and JS handles the interactivity and functionality.

We needed to modify the JS code to handle touch events instead of mouse events. This likely involved rewriting or modifying event listeners to recognize touch gestures such as taps, swipes, and pinches, and to collect the touch-related data we needed for the machine learning model.

Overall, your data acquisition process involved modifying existing mobile web applications, collecting detailed touch-related data, and exporting the data in a compatible format. These steps provide a strong foundation for training a machine learning model to recognize and predict touch owners.

Note that at the date this report was written the application was not entirely finished and was just a prototype after up to 8 days editing code due to issues that will be described in the last section of this document.

II. Preprocessing

After data acquisition we had to represent data in a suitable form for training a machine learning model. For this purpose we wrote a python script that open and read each downloaded .json files and extract fields we was interested in from them such as positions (list of 2 items' lists where the first is Position.X and the second Position.Y), speeds (list of Euclidean speeds calculated for each new position recorded and divided each time by the time elapsed since the touchstart event listener has been triggered), directions (list of direction/angle of each touch with regard to the previous in radians) and timestamps that we used to compute the duration of each touch.
Note that positions have been split into two new lists position.X and position.Y.

And then in order to obtain the features that will be used to train the model we calculated the mean, standard derivation, min, max and range of each field or lists (Position.X, Position.Y, speeds, directions) and appended the duration of the touch. This way we got a new representation of our data that is a numpy.array of shape (21,) which is a very good representation for training a Deep Learning model.

Data has then been labeled and saved with **Pyhton.pickle** .

NB: Label was binary values (0, 1)

AI Training & Evaluation

I. Creation & Training

We decided to go for a Keras model.

Keras is a Python library for deep learning that wraps the efficient numerical libraries Tensorflow and Theano. Keras allows you to quickly and simply design and train neural networks and deep learning models.

So we create a quite simple Keras Sequential model.

The weights are initialized using a small Gaussian random number. The Rectifier activation function is used. The output layer contains a single neuron in order to make predictions. It uses the sigmoid activation function in order to produce a probability output in the range of 0 to 1 that can easily and automatically be converted to crisp class values.

Finally, you will use the logarithmic loss function (cross-entropy) during training, the preferred loss function for binary classification problems. The model also uses the efficient Adam optimizer for gradient descent, and accuracy metrics will be collected when the model is trained.

The model had just one hidden layer of 10 neurons. So the Model topology can be summarized by [21, 10, 1]. It had just 693 parameters

II. Evaluation and saving

We firstly evaluated couple model topologies using stratified cross validation in the scikit-learn framework.
The idea was to use this step as a test consisting of training for a short while and evaluate them in order to choose the best model to train for a longer time.

The stratified cross validation (StratifiedKFold) is a resampling technique that will provide an estimate of the performance of the model. It does this by splitting the data into k-parts and training the model on all parts except one, which is held out as a test set to evaluate the performance of the model.

Due to a lack of time we nearly skipped this step. We have been really surprised by the performance of the model even without any hidden layer that scored 83% accuracy.

After further training of just 1000 epochs the topology [21, 10, 1] scored up to 90% accuracy and it's the model we have saved at the time of writing this document (May 13, 2023).

We then saved the pre-trained models ready for inference, test or further training in different formats:

- In a folder

The SavedModel format is a directory containing a protobuf binary and a TensorFlow checkpoint. The first approach used to save the model

- In a TF.js Layers format

We saved the model in TF.js Layers format after a brief test to use it with the app later

Model Deployment

This is the actual step we are working on.

We already exported the model to Tensorflow.js layers format and we are trying to integrate it with the web application used to collect data but we actually face some problems and import issues related to TFJS and dom-events. More details in issues section

Issues

We had a delay of 2 weeks to do this partial report.
We encountered some issues, most of them have been solved or were more programming errors than real issues or bugs.

In this section we are going to talk about two issues / bugs that have not been corrected yet.

I. Mouseenter => Touchenter events

Finishing the UI of the web app involves finding alternatives to mouse events in touch events. In A part of the code we must do this conversion but it seems like the touchenter event doesn't really act like the mouseenter one does for mouse events.

The mouseenter event is triggered whenever the mouse enters the area of an element in the HTML while the touchenter event is triggered when the touch starts on the element like the touchstart event that is often used at the place of mouseenter. We are actually working on a solution to fix that. If needed we envisage to rewrite the code to not be dependent of this events anymore

II. Importing Tensorflow.js in a module that use dom-events

We noticed that importing Tensorflow.js in a JavaScript module that uses dom-events will prevent these events from working.

So we are working on that issue too in order to integrate the model in JavaScript and use it in the browser.

For this purpose Carlos proposed to help us, knowing he worked around this problem some years ago with another version of Tensorflow.js. His help will be precious for us and we may be able to quickly solve or work around this problem.

Summary & Notes

Summarizing this document was presenting an ongoing little research project held by five students in Mali.

With results described here we think that this project has a big potential and would like to get people interested in it.

We hope that this can be a starting point for a big progress in AI and CyberSecurity fields and IT globally.
Hoping we could participate in the future of technology with this project.

A final report will be done once the project is finished, there is nothing spectacular for now but hope it will be.

You can get all codes written to this date on this github and contacts authors with the following links:

Kanté Younouss Koly: email

Haïdara Mohamed: email, Facebook, linkedin,

Diabaté Maïmouna: email

Diarra Yacouba: email, Facebook, linkedin, twitter, github, Stackoverflow

Maïga Maïmouna Henna: email, twitter, linkedin